Boolean Logic Simulator in C++

Software Architecture Document

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 04/08/2024 | 1.0 | Initial Template Filled Out | Kemar Wilson |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 4

1.5 Overview 4

2. Architectural Representation 4

3. Architectural Goals and Constraints 4

4. Use-Case View 4

4.1 Use-Case Realizations 5

5. Logical View 5

5.1 Overview 5

5.2 Architecturally Significant Design Packages 5

6. Interface Description 5

7. Size and Performance 5

8. Quality 5

Software Architecture Document

# Introduction

HL Group aims to develop a C++ Boolean Logic Stimulator software application that enables users to perform logical operations based on Boolean algebra. This Software Architecture Document provides a comprehensive overview of the architectural design and structure of the Boolean Logic Calculator software. The document outlines the purpose, scope, definitions, acronyms, and abbreviations used throughout the software development process. By detailing the architecture and design principles, this document serves as a reference guide for developers, stakeholders, and users involved in the creation and utilization of the Boolean Logic Calculator software.

## Purpose

# The Software Architecture Document for the Boolean Logic Simulator in C++ serves as a fundamental guide that provides a comprehensive architectural overview of the system. It is designed to capture and convey the significant architectural decisions made during the design and development of the Boolean Logic Simulator. This document utilizes various architectural views to depict different aspects of the system, ensuring a holistic understanding of the architecture.

# Role and Structure of the Document: The primary role of this Software Architecture Document is to outline the architectural design and decisions behind the Boolean Logic Simulator project. It serves as a roadmap for developers, architects, and stakeholders to understand the system's design principles, components, interactions, and constraints. The document is structured to present clear and detailed insights into the system's architecture, emphasizing key design choices and their implications.

# Audience and Usage: The document is intended for a diverse set of audiences involved in the project, including software architects, developers, project managers, quality assurance teams, and stakeholders. Software architects will use this document to define the system's high-level structure, interfaces, and behavior, guiding development efforts. Developers will refer to it for detailed technical information on the system's components and interactions. Project managers can leverage the document to assess project progress and ensure alignment with architectural guidelines. Quality assurance teams will utilize it to define test strategies aligned with the system's architecture. Stakeholders will gain insights into the system's architecture and understand how it aligns with business objectives.

## Scope

The Software Architecture Document for the Boolean Logic Simulator in C++ provides a comprehensive overview of the system's architecture, capturing significant design decisions. It influences system design, development efforts, interactions, scalability, performance, maintenance, and evolution. The document is intended for architects, developers, project managers, quality assurance teams, and stakeholders, guiding them in understanding and utilizing the system's architecture to ensure successful project outcomes.

## Definitions, Acronyms, and Abbreviations

Project/Program: Boolean Logic Simulator in C++ as described by the SDS. See *Section 1.4 References for SDS*

SA/This document: Software Architecture Document. See *Section 1.4 References for SRS*

SDP: Software Development Plan. See *Section 1.4 References for SDP*

SRS: Software Requirements Specifications. See *Section 1.4 References for SRS*

## References

SRS: Accessible at <https://github.com/KNEternity/348HL/tree/main/docs>

SDP: Accessible at <https://github.com/KNEternity/348HL/tree/main/docs>

SA: Accessible at <https://github.com/KNEternity/348HL/tree/main/docs>

## Overview

[This subsection describes what the rest of the **Software Architecture Document** contains and explains how the **Software Architecture Document** is organized.]

# Architectural Representation

[This section describes what software architecture is for the current system, and how it is represented. It enumerates the views that are necessary, and for each view, explains what types of model elements it contains.]

# Architectural Goals and Constraints

[This section describes the software requirements and objectives that have some significant impact on the architecture; for example, safety, security, privacy, use of an off-the-shelf product, portability, distribution, and reuse. It also captures the special constraints that may apply: design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.]

# Use-Case View

[This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage—they exercise many architectural elements or if they stress or illustrate a specific, delicate point of the architecture.]

## Use-Case Realizations

[This section illustrates how the software actually works by giving a few selected use-case (or scenario) realizations, and explains how the various design model elements contribute to their functionality. If a Use-Case Realization Document is available, refer to it in this section.]

# Logical View

[This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. You should introduce architecturally significant classes and describe their responsibilities, as well as a few very important relationships, operations, and attributes.]

## Overview

[This subsection describes the overall decomposition of the design model in terms of its package hierarchy and layers.]

## Architecturally Significant Design Modules or Packages

[For each significant package, include a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.

For each significant class in the package, include its name, brief description, and, optionally, a description of some of its major responsibilities, operations, and attributes.]

# Interface Description

[A description of the major entity interfaces, including screen formats, valid inputs, and resulting outputs. If a User-Interface Prototype Document is available, refer to it in this section]

# Size and Performance

[A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.]

# Quality

[A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on. If these characteristics have special significance, such as safety, security or privacy implications, they must be clearly delineated.]